Meltem Tutar
CS230 Final Report

Sentiment Analysis on News and Politics Text

Introduction: We are performing sentiment analysis on comments that are in the political/news domain. Sentiment analysis is the computational study of people's opinions towards entities such as products, services, organizations, individuals, issues, events or topics (Liu, 2015). Determining public opinion regarding current events could be very useful for political bodies deciding on resource allocation and future policies. **Git URL: https://github.com/mtutar-stanford/sentimentAnalysis**

Dataset: The dataset is from twitter comments on the 2016 GOP debate. It originally came from Crowdflower's Data for Everyone library, but we downloaded it from Kaggle. Contributors labeled if the tweet was relevant, which candidate was mentioned, what subject was mentioned, and what the sentiment was for a given tweet. Sentiment options are Neutral, Positive, or Negative. There are 13,874 tweets in total.

Figure 1: A histogram of the number of words in each training example

We have removed all columns besides [sentiment] and [text], which is the tweet classification and text respectively. We also further cleaned the dataset by removing hashtags ('#'), mentions ('@'), retweets ('RT'), and re ('Re'), hyperlinks, and whitespace. We performed a 75%, 15%, 10% split. This gives 10405, 2081, and 1387 for training, dev, and test sets respectively. The average number of words are displayed in Figure 1.

Previous Work: Sentiment analysis is challenging given that deciphering sentiment is subjective, and that historically there weren't many available sources of labelled data. With the introduction of websites that provide organized collections of opinionated documents, such as Epinions and newswire, early researchers in the 1990's began applying supervised and unsupervised machine learning algorithms to predict sentiment of texts. In the supervised setting, models such as as Naive Bayes, Support Vectors machines, and maximum entropy-based classification are explored in the literature. Different feature selection schemes have been utilized for representing text in numerical forms for these supervised approaches. These include term frequency–inverse document frequency (tf-idf), Term presence, and n-grams. [1] Other methods for text representation include creating word embeddings where words or phrases from the vocabulary are mapped to vectors. Unsupervised methods rely more upon detecting patterns in sentence structure and word usage, and include methods utilizing POS tagging and sentiment lexicons [2]. We chose the baseline model as a classical supervised approach, and implemented a multinomial Naive-Bayes using tf-idf to represent words in sentences.

With the boom of web platforms such as Twitter, Reddit, blogs, and forums, there has been an increase in data available for sentiment analysis, and a shift towards the application of neural

networks, which traditionally perform better on large datasets. Neural networks can be categorized into feedforward and recurrent (RNN). RNN's are most common in NLP literature because predictions from previous inputs affect next outputs, which makes it suitable for data having a sequential nature. Long short-term memory models (LSTM) are a variant of RNN, which allows for learning across longer sentences by decreasing the vanishing and exploding gradients problem that traditional RNN's have. Gated Recurrent Unit's (GRU) are an alternative to LSTM's that are less complex but achieve similar performance [3]. In these models, text is represented as word embeddings, where the embeddings may be learned by the model itself or begin with pretrained vectors learned from probabilistic models or neural networks. Examples of the many available embeddings online include word2Vec, GloVe, and fastText [4]. To improve upon the baseline model, we chose to build a GRU utilizing pretrained GloVe word embeddings.

Convolutional Neural Networks (CNN's) are a variant of feed forward neural networks, and in recent years have begun to be utilized in sentiment classification tasks. CNN's learn features of sentences by projecting the vector representation of sentences onto lower dimensions and creating a feature mapping. Filters of size n corresponds to learning feature maps from n-grams. Kim (2014) analyzed different variants of CNN's, such as static, non-static, and multichannel and reported that a simple CNN with one layer of convolution and little tuning of hyper parameters performed remarkably well for sentiment analysis. Motivated by this example, we decided to also build a CNN model with max pooling over time and pretrained word embeddings. [5]

Methods: We implemented three models 1) multinomial naïve-bayes as a baseline 2) GRU with GloVe pretrained word embeddings 3) CNN model with max-pooling over time. The motivation for each is described in the 'Previous Work' section.

Multinomial Naïve-Bayes with TF-IDF

Before we apply the multinomial naïve bayes algorithim, we apply TF-IDF, which creates a vector of the text in the example. Each entry in the vector are the counts of occurrences of our vocabulary multiplied by weights determined by how frequently the word appears in other examples, and how many words are in the current example. The higher the number of words in the example, and the more it appears in other examples, the lower the score will be. The TF-IDF is a measure of how important a word is for an example in determining sentiment. We utilized SKlearn's CountVectorizer and TfidfTransformer packages to perform this process.

Naïve bayes classifers attempt to model the joint probability distribution, which is rewritten using naïve assumptions as in Equation 1.

$$P(C = c, X_1 = x_1, \cdots, X_v = x_v) = P(C = c) \prod_{1}^{V} P(X_j = x | \, C = c)$$

Equation 1: Joint probability being estimated with Naïve Bayes

$P(X_j = x | \, C = c)$ and $P(C = c)$ are estimated from the training data using the maximum likelihood estimates (MLE) of equation 1. They are written in equation 2.

$$P(X_j = x|\ C = c) = \frac{N_{ci}+\alpha}{N_c+\alpha n} \qquad P(\ C = c) = \frac{NC}{n}$$

Equation 2: MLE estimates of probabilities calculated from the training set
N_{ci}: number of times feature i appears in a sample of class c in the training set
N_c: total count of all occurences of all words in class c
NC: number of examples of type c, n: number of training examples

With these estimates, the class that maximizes the joint distribution for a test example is predicted as portrayed in equation 3.

$$\arg \max_{y\in\{1\ldots k\}} p(y, x_1 \ldots x_d) = \arg \max_{y\in\{1\ldots k\}} \left(q(y) \prod_{j=1}^{d} q_j(x_j|y) \right)$$

Equation 3: making prediction on a test set with multinomial naïve bayes

GRU with GloVe word embeddings

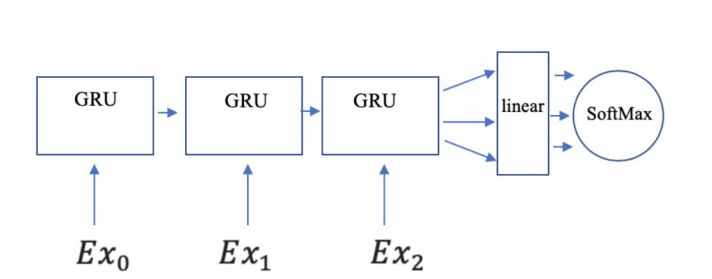

Figure 2: A high level view of GRU architecture, where each GRU unit has an update gate and reset gate and E represents the embedding matrix, and x represents words as tokens

We've used torch text, spaCy, and Pytorch to create the GRU and CNN models. Torchtext allows easy tokenization of sentences using spaCy, creation of a vocabulary, and generation of padded batches.

For the milestone, we experimented with different features such as multi-layers, regularization, and different activation functions for the RNN model. The variant that resulted in the highest accuracy was a 2-layer LSTM with bidirectionality, dropout of 0.5, hidden dimension size of 256, and embedding size of 100. Later we implemented a GRU with the same hyperparameters, which produced similar results to the LSTM, and decided to continue with the GRU given it is less complex. Each GRU unit has an update and reset gate, which allows it to alter the inputs that it receives from previous units and avoid the exploding/vanishing gradient problem.

After the GRU module, we subsequently apply a linear layer, and SoftMax to predict one of the 3 classes of sentiment. A high-level overview of the architecture is shown in Figure 2. We used adam's optimization algorithm and are applying SoftMax and subsequently minimizing cross entropy loss as shown.

$$CE\ Loss = -\sum_{c=1}^{3} y_{o,c} \log\left(p_{o,c}\right)$$

Equation 2: Cross entropy loss with SoftMax
y: binary indicator $(0\ or\ 1)$ if class label c is the correct classification for observation o
p: probability observation o is of class c, calculated by taking the softmax of the linear outputEquation 1: Cross

CNN with max pooling over time

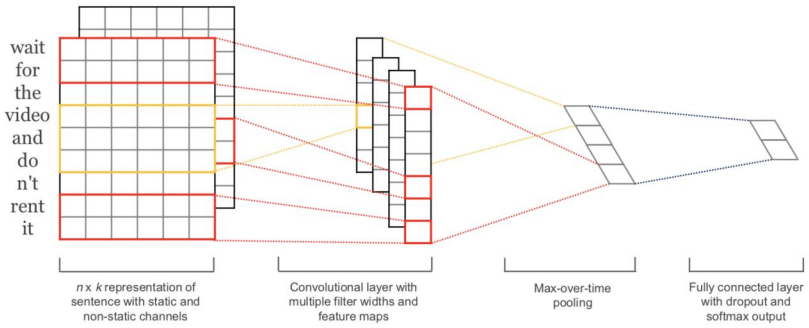

Figure 3: A high level overview of The CNN with max over time pooling architecture

The pretrained word embeddings were still utilized in this model. 100 Filters of size 2,3, and 5 were applied. We padded examples to have a minimum of 5 words.

The size of the filter dictates the window of words that it analyzes and can be equivalent to creating n-grams. The appearance of certain bi-grams, tri-grams, and n-grams within the text may be indicative of final sentiment.

Subsequently, a max-over-time pooling layer is applied, which is simply the max of the output of each filter of all the sizes, hence it's dimension is 300 (3x100). The max-over-time pooling layer is applied so that inputs of variables dimensions have the same size output. Finally, it is passed through a linear layer so that the output has dimension of 3. Adam's optimization algorithm is utilized, and we are still applying SoftMax and minimizing cross entropy loss as shown in equation 2.

Results:

Neural Network Models	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
Baseline: Multinomial Naïve-Bayes, TF-IDF	--	0.643	--	0.689
GRU with bidirectionality, 0.5 dropout, 2 layers, 256 hidden dimension, 100 embedding dimensions	0.503	0.798	0.867	0.651
CNN with max pooling over time, 100 filters of size 2,3, and 5.	0.559	0.773	0.784	0.681

The training and validation accuracies models after training for 30 epochs. Accuracy is the proportion that are correctly predicted, with no difference between false positives and false negatives. (Table 1).

GRU Training/Validation Curves

Figure 3: GRU accuracy plotted over 30 epochs

Figure 4: GRU cross entropy loss plotted over 30 epochs

Discussion: We weren't able to make improvement from the baseline on the validation set, though we did have higher accuracy on the training set. It seems that the CNN/GRU modes are not generalizing well. The CNN loss curve is similar to the GRU curve. The large decrease in training error and eventual increase in validation error may indicate there is an overfitting problem. Dropout is applied, and various other regularization methods were also experimented with, without much change in the situation. I ran the CNN model that was trained on 10 epochs on the test set to get an unbiased estimate of error. The accuracy and loss were similar to the validation set, with a loss of 0.785 and accuracy of 0.68.

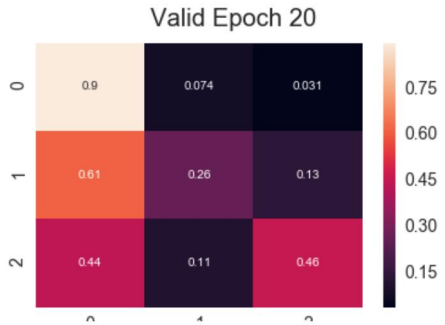

Figure 5: Confusion Matric after training the GRU model for 20 epochs

To understand where the error is propagating from, we also plotted confusion matrices. It is apparent that most of the errors are being made on examples from class 1 and 2. There is a slight imbalance in class, with 60% of the training examples from class 0.

Future Work: We believe that addressing the imbalances of classes may allow the CNN/GRU models to generalize better. We are surprised that the deep learning models were not able to do substantially better than the naïve bayes. We believe we need to fundamentally alter the models by experimenting with different objective functions to optimize, introducing new features such as POS tagging, and possibly learning from other training sets.

References

1. Pang, Bo, and Lillian Lee. "Opinion mining and sentiment analysis." Foundations and Trends® in Information Retrieval2.1–2 (2008): 1-135.
2. Turney, Peter D. "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews." Proceedings of the 40th annual meeting on association for computational linguistics. Association for Computational Linguistics, 2002.
3. Zhang, Lei, Shuai Wang, and Bing Liu. "Deep learning for sentiment analysis: A survey." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 8.4 (2018): e1253.
4. Berardi, Giacomo, Andrea Esuli, and Diego Marcheggiani. "Word Embeddings Go to Italy: A Comparison of Models and Training Datasets." IIR. 2015.
5. Kim, Yoon. "Convolutional neural networks for sentence classification." arXiv preprint arXiv:1408.5882 (2014).